TURBO-FREEZER XL/XE 2005

Version 0.90 as of 05.01.2006

Contents

Preface by the designer of the new Freezer

The project to design a new Turbo-Freezer was started early 2004, when Bernhard Engl gave permission to the ABBUC to remake his products (The turbo 1050 and the Turbo-Freezer). To me, the Turbo-Freezer has always been the most interesting piece of hardware extension that has ever been available for the Atari and before starting the project, I had already studied and analised how it worked. That's why it was pretty obvious that I would engage in rebuilding the Turbo-Freezer. There were more participants in the team: Bernhard Pahl, Florian Dingler, Torsten Schall, Guus Assmann, Frank Schröder and towards the end also Wolfram Fisher.
So I would like to express big thanks to all of you. The project wouldn't have com this far without You!! And some special thanks to Guus, for the many ideas for extensions, designing the PCB's and producing the proto-types. Thanks as well to Bernhard Engl for his tips and the detailed information for the original Turbo-Freezer. All other team members, thanks for the support and the testing!
The original plan was to remake the Freezer, using modern parts. At this point, I never imagined that the result would be such extensive and powerfull improvements! During the project, both hardware and software have been gradually extended and improved up to a point where only the basic (and genius) principle idea of the freezer was left. All else had been changed.
Some of the extension came by as pure coincidence. For example the cartridge emulation: As a 16K Eprom was much more expensive, Guus suggested using a Flash-Rom. To enable programming this Flash from the Atari, it was needed to map the chip into the memory-map of the Atari. This formed the basis for the cartridge emulation :-)
Likewise with the Ram memory: The original 2K Ram was hard to get and also quite expensive. So the first step was to upgrade to a 32K chip and also use 128K Flash-rom. The next step was to go to 128K Ram and this made it possible to put a memory-snapshot into this Ram. And the Flash was increased to 512K. To enable this, the logic had to be improved. And a bigger logic chip, made room for the emulation of the 16K OSS-modules and the Sparta-Dos module.
With the current state of the logic chip's internals, there's no room left and we're all content to have gotten the maximum out of the parts that were used. For the Software, there's some more ideas for extensions, but that all I'll say for now.
During the development process, I've constantly kept in mind that I wanted to create an open basis for further developments. For this reason, all design information is freely available and there's even a J-Tag interface implemented so the logic can be reprogrammed. This will help anyone who wants to expand the Freezer or even make something completely new with it.
All relevant information and updates for the Turbo-Freezer XL/XE 2005 are available on the internet at http://turbofreezer.horus.com There's also data on how to alter the logic of the Turbo-Freezer. For technical questions, write an E-Mail to me at hias@horus.com
The many functions of the Turbo-freezer have not only convinced the team members, but also many members of the ABBUC.
In their Hardware Contest 2005, it was awarded a very convincing first prize.
I wish you a lot of fun using the Turbo-Freezer and keep loyal to the Atari 8-bit computers for many years to come.

Salzburg, December 2005
Matthias Reichl.

Preface by the developer of the original Freezer (2005)

As a designer, the best recognition of achievement one can get, is that a once successfull product is continued and enhanced by a competent sucsersor. However, when this happens after 16 years and for a home-computer product, it's down right sensational and means that the product has become a cult-status. While using original concept of the 1980's, Matthias Reichl has made a real masterpiece of the TURBO-FREEZER XL/XE 2005.
And the application of modern micro-chips has lead to a great extension of the functionality. He even managed to remove some minor bugs that were known, but couldn't be prevented using the electronics of that time, without making the product economically impossible because of the higher chip-count.
The improvements and enhancement, especially the ingenious cartridge emulation that was not present in the original, make the TURBO-FREEZER XL/XE not only to the best and most perfect Freezer for the 8-bit Atari that ever existed, but also is a multi-talent, having the potential to become a legendary cult product.
At this point in time, it's a high-light of a technical development that started more than 20 years ago. By now, my notebook has turned yellow and the first sketches of the concept date back to the autom of 1984 (Finding this did even surprise me).
To experience a vastly improved, enhanced and rejuvenated new series of a computer extension product, is not only a miracle, but also a special joy an honor to me, so I wish all proud owners of the new TURBOFREEZER XL/XE a lot of pleasure using it.

Zurich, November 2005
Bernhard Engl

Preface by the developer of the original Freezer (1987)

One year development time, hundreds of user IC's, 5300 lines of assembly and cost in the 5 number range, it's there at last: The TURBO-FREEZER XL. It's not only one of it's kind, it's the one enhancement that the Freak needs to get the optimum and maximum usage out of their Atari.
Apart from the Freezer, there's also room for an Oldrunner and 256 Kbytes of Ram that upgrades the 800XL to 320K. Thanks to the use of 3 ASIC's, containing the equivalent of over 40 TTL-Circuits and the fact that it has just been managed to avoid a very costly multi-layer PCB, the price disables all competition.
On top of this, thanks to the very powerful memory management that is needed for the freezer, the computer doesn't need to be opened. As with the hardware, also the software is a careful balance in functionality, like Freezer, mini-dos and a debugger. Combined with the hardware, this power-tool gives an unprecedented possibility of control over programs to the user
That's why anyone who's seen the Turbo-Freezer XL in action, will want to have one as well. The thrill of having total control over the computer, even though the programmer did not wish this, is well worth the price. And then there's also the money that has not been spent on the parts that would be needed to get the same functions that are present in the Turbo-Freezer XL
More about this in the description of the particular functions! I wish every proud owner lots of fun with the most fascinating product available for the Atari.

Munich, May 1987
Bernhard Engl

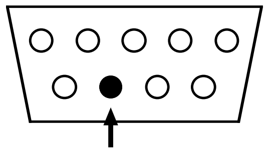
1 Installation

The Turbo Freezer is connected to the parallel bus. The power supply is through this bus, for the Atari 600XL and the older 800XL and Atari XE, or by means of connecting the extra cable to the joystick-port for some XL models. There's no need to open the Atari or make any solder connections. But experienced user with the right tools, may want to put some extras into the computer.

1.1 Plugging the Freezer into the Parallel Bus Port

1.1.1 ATARI 600 XL (mind. 64kRAM) und ATARI 800 XL

Simply remove the cover of the bus at the back side of the Atari and bend the metal fingers a little to the outside. Now gently plug in the Freezer. Connect the cable to pin 7 of joystick port 2. It is marked in the adjacent drawing.



Important: Do not connect the cable when the Atari os powered on, as this may damage the Freezer and/or the Atari. If the cable is removed (by accident) during the operation, switch of the Atari before reconnecting the cable. (Remark 2005: Although Lattice, the company that produces the GAL's, claims "the Gal will not latch-up under any conditions" in the data-sheet, a lot of fried freezer proved otherwise. The modern chips are more robust and don't use the old charge pumps anymore to bias the substrate. So nowadays, it's hard to destroy a C-Mos chip. But even so, better safe than sorry. Thus the warning still stands B.E.)

1.1.2 ATARI 130 XE

Plug the Freezer into the back of the Atari. The switches and module connector should be on top.

1.2 First Power Up

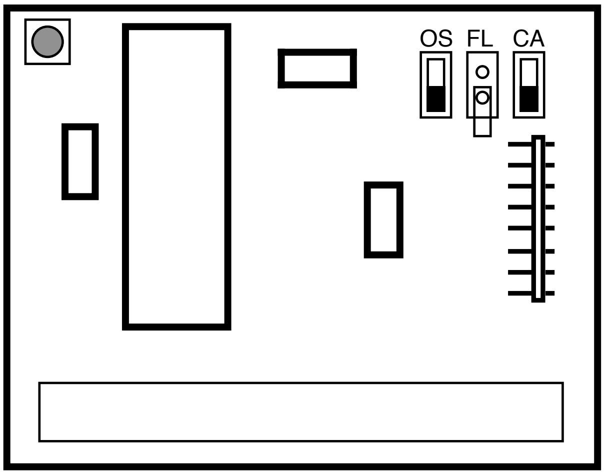
1.2.1 Step 1: Setting the basic configuration

Set the Freezer in basic configuration:

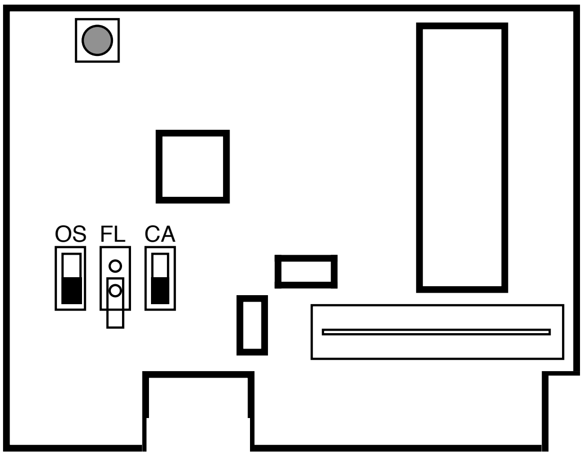
- Oldrunner "OS" switch off.
- Flash "FL" Off, open the jumper or put it on 1 pin.
- Cartridge Emulation "CA" off.

For the XL model, this means move all switches to the middle of the PCB and open the jumper between the switches.

For the XE model, this means all switches toward the Atari and open the jumper between the switches.



Freezer XL



Freezer XE

1.2.2 Power up with the Freezer

Just switch on the Atari. If there's no "READY" on the screen after the normal pause, but strange symptoms instead (System crash, black screen, smoke) switch of at once and check for faults.
If the oldrunner is active, "ATARI COMPURE - MEMO PAD" will appear instead of "READY" If the Cartridge emulation is active, the screen will be green and display "TURBO CARTRIDGE"
In this case, look at chapter 1.2.1 to set the basic configuration.

If there's "READY" on the screen, just push the red FREEZE-Button (To the left top of the PCB) to get the FREEZER-Menu.
In case nothing happens, replug the Freezer, check the cable (If it's there) or the Freezer is defective.

From the freezer menu, push the "space-bar" This will get basic back. If this is not the case, there's an error.

But if all went well up to this point, the Freezer is ready for use.

1.2.3 Is the Cable needed? (Only for the Freezer XL)

To test if the cable is needed, go to the Freezer menu and remove the cable. (Do not put it back with the Atari switched on!) If all keeps working, the cable is not needed. In this case, don't cut it, but make a neat roll of it. This enables future use on an Atari that has no power to its parallel bus.

1.2.4 If the Freezer is defective

Before returning a defective Freezer, it’s advisable to try and contact someone by snail- or E-Mail. Give a description of the symptoms. The Freezer XL/XE has been extensively tested and therefore it’s almost impossible that it is defective. The more likely cause is a connection error or a defect in the Atari. So the best cause of action is to exchange some information prior to the more drastic action of returning the Freezer. This prevents cost and waiting.

(Hinweis 2005: das war das schon damals bekannte Timingproblem mit manchen ATARIs, bei dem die ATARI-eigene Spezifikation des parallelen Bus-Timings krass verletzt wurde. Wir haben es behoben, indem wir uns den ATARI einsenden ließen und die CPU oder den ANTIC kostenlos gewechselt haben. Die Chips hat uns ATARI Deutschland ebenso kostenlos zur Verfügung gestellt.

Die neuen Freezer haben Latches an den kritischen Adressleitungen, so daß dies kein Problem mehr sein sollte, auch wenn die damaligen schlechten Chips im Spiel wären. B.E.)

1.2.5 Creating a Turbo-Freezer system disk

Once the preliminary tests were successful, it’s time to create the system disk. After production and testing, the software to create the disk is programmed into the Freezer. This disk will contain the flash-software and the Freezer software. With these programs, the Freezer may be restored to its basic configuration at any time.

To make the system disk, do the following:

1. Switch of the Atari, connect a disk drive and activate the cartridge emulation. (For XL: CA switch up, for XE: CA switch away from the Atari) Power up the drive. And now the Atari.
2. After powering the Atari, the Cartridge emulation menu should be displayed. All default settings are correct, so just press [RETURN]
3. Now, the “Turbo-Freezer System disk writer software” is activated. Put an empty disk in the drive (D1:) and press [ENTER] The program will now format the disk (in enhanced density) and writes the TURBO-FREEZER software on it. If there’s an error during this process, (Like in case of a defective disk) the program will display “ERROR” and a new disk should be put in the drive. Just restart the program (enter [Y] at “restart program”) For extra security, make a second disk, write protect it and put it in a safe place.
4. After a successful creation of the disk, switch of the Atari and put the “CA” switch in the initial position.

If all goes bad and the disk creation fails or the disk gets corrupted or rewritten, (Yes, this may happen) then the data can be found on the internet. http://turbofreezer.horus.com/ has an ATR-disk image ready for download.

1.3 For Experts in soldering: Internal power supply

If you have an Atari 800XL without power to the parallel bus, this can be added, provided that you have enough knowledge and skill. If in doubt, it's also possible to attach a longer wire to the PCB and connect this to the power supply inside the Atari. This leaves the joystick port for its intended use!

Warning: This bit is really for experts only. If you're not the expert, you may try to find one. It's no use to find a service provider for this (Very expensive), nor is it wise to kill your Atari over this.

In order to disappoint beginners, the description of the job is kept to a minimum. For the expert however, this will be quite enough. If you've found an expert to help and you notice that he's having a hard time to figure out the information, there's still a chance to stop this obvious amateur.

Here's the description: The connection is best done to L1, on the point where a trace goes inside the shield. This coil L1 is in close proximity to the power switch. From this point, you either connect the Freezer power cable, or make a power connection to the parallel bus. This is done to pins 47 and 48 of the bus. (Those are next to the pins 49 and 50 at the edge of the connector, on both the top and bottom of the PCB) This is difficult, because the Atari has to be taken apart and it's a tedious job to solder on the end of the contact surface, without having the solder flow over the total (gold plated) contact. It's obvious that this pollution has to be prevented.

1.4 For Experts in soldering: Installing a System Reset button.

Using the Old-runner and to be able to interrupt programs that don't allow this, it helps to have a button that activates an NMI. (Non Maskable Interrupt). The button is connected so it will pull pin 6 of the Antic to ground. The best way to do this is with a connection to resistor R31.

Warning: See also 1.3. In addition, there's some mechanical work to be done as well, to place the button. For this, skill and tools are needed, to ensure that it will look decent.

1.5 Compatibility to 1MB ramdisks

The Turbo-Freezer uses the Refresh-line to stop the Atari and hence there are some compatibility problems with Ram-extensions that also use this. A refresh based on the Antic refresh will have this problem. These are mainly the Newell 1MB. Most 256K extensions, and also the 130XE don't have this problem.

Using two additional parts (A small signal Schottky-diode, like the BAT85 and a 4K7 resistor) will solve this problem.

Warning: See point 1.3

After opening the Atari, bend up Antic pin 8. Remove the cable that connects this pin to the ramdisk.

If the Antic is in a socket, remove it and bend up pin 8. Attention: don't bend it to sharp or it will break. Just bend it far enough so it will stay clear of the socket when the IC is put back in.

If the chip is soldered in the board, use a pair of fine wire cutters to cut the pin just above the PCB and bend it up a bit.

Now, solder the diode between pin 8 and the socket or PCB, to where the pin used to be.
The cathode (Ring on the housing) should be connected to Antic, the anode to the socket or PCB.
The easiest way is to solder the diode to the Antic chip and use a cable to connect the diode to the bottom of the PCB.
One side of the resistor is connected to the Anode of the diode, the other side to +5V. This can be found on pin 21 of the Antic.
The cable from the ramdisk should now be connected to pin 8 of Antic again.

2 Using the Freezer

Did you ever have an excellent game, but were annoyed by the fact that it lacks a way to pause it? Or the kind that lets you start all over again, when all lives are lost, and you have to play forever to get to the point that you reached before? Or a game that lets you play a very long time, to reach the levels with the real action? In these cases, the Freezer is just what you need:
You can Freeze any program at any time, store it to any media you like and reload it later, to continue at the point you've left of before. And do this as often as you like, so it's no problem to use hundreds of lives to master a part, where there were only one or two lives left.

To have real fun with the Freezer, it should be available at all times, have resident software (So no need for clumsy Diskette or Cassette boot processes) and work fully automatic in seconds. To achieve this, there have been no economising actions, that would have been possible, like having the user restore part of the data in hardware registers or enter some none resident software.
The Turbo Freezer has been designed to offer the best possible solution, with the current technical means and not rely on dubious measures that can be observed on other computer systems. And this effort has become rewarded, like the product will show you. And once you've seen the Freezers on other systems, one can only look in amazement when the Turbo-Freezer is operated at it's astonishing speed. For example, to save a current game to ram-disk and continue playing three seconds later, only 3 key-stokes are needed. The same applies to restoring from the Ram-disk a couple of seconds later. And the C-64 user that used to laugh at the Atari for not having a Freezer, will now have his mouth fall open and crack his jaw on the floor!

But the Freezer can do much more. It is possible to convert from Cassette to Diskette. Cassette owners can take their beloved software and store it on their new Diskette unit. And diskette user can use the programs without tedious loading or save money by buying the Cassette versions.

And there is one more point: The Turbo-freezer is the first freezer with a Dos and debugger built in, so this is available all the time, without damaging the frozen software! And a "disk full" error is no problem in any program, even if it has no DOS features available. And a serious Bug, that cannot be found without a deeper look into the system hardware registers is no problem any more.

2.1 Freezing and resuming a program

After pressing the red button, the freezer takes control during the next interrupt cycle and saves the current system status. The main menu of the freezer grants access to the functions of the device. And the program can be resumed at exactly the same point from which it was frozen, by pressing the [SPACE] bar on the keyboard. After loading a frozen program, it's automatically resumed. Pressing the [RESET] will interrupt the freezer actions in a controlled manner and execute a cold boot.

Freeze	Freezerbutton
Resume	`[SPACE]`
Coldstart	`[RESET]` while in Freezer menu

If the resume is initiated with [SHIFT] + [SPACE], the player missile collision registers are cleared before the resume gives control to the program. This function will not be needed in normal circumstances, but with some games it makes sure that no "game-lives" are lost after the resume. When loading a frozen program, this function is also available. Just press [SHIFT] + [E] , [X] or [C]

A program that is loaded using [CONTROL] + [E] , [X] or [C] will not resume directly after loading. The menu stays in control and some memory locations can be modifies before resuming. This feature is particularly practical when looking for the memory location for the amount of energy / lives. Just freeze the program, save it, change a memory location and resume. If the location was not the right one, reload using [CONTROL] + [E] , [X] ,[C] and try again on the next location. Now, the tests have the same starting point and its not a question how to undo previous changes.

[E]	Start a program from an external device (Cassette or diskette)
[SHIFT] + [E]	Start a program from an external device (Cassette or diskette) but clear the player / missile collision registers first
[CONTROL] + [E]	Load a program, like above, but don't start it yet.
[X]	Start a program from Ramdisk
[SHIFT] + [X]	Start a program from Ramdisk but clear the player / missile collision registers first
[CONRTOL] + [X]	Load the program from Ramdisk and don't start it
[C]	Start a program from the freezer ram
[SHIFT] + [C]	Start a program from the freezer ram but clear the player / missile collision registers first
[CONRTOL] + [C]	Load the program from freezer ram and don't start it

In theory, any program may be frozen and resumed at any place, even during the cassette or diskette operations. However, this is not wise, as this may cause the function to remain incomplete. And it may be possible for the OS to resume, but this may not be the case for the device that was interrupted. For the cassette this is sure not to work, as the unit is too "stupid" for this. To conclude, it's better to only freeze a program that is not accessing a device.

In some rare cases, pushing the freeze-button may have no effect at all, the program just keeps running. The secret is that such a program doesn't use any interrupts. This may be the case for simple programs, such as converted APPLE programs that don't use any of the ATARI specific

features. The solution to this problem is to install a system reset key. (See Chapter 1.4 about this) The interrupt caused by this key cannot be masked, so there's no defence against the TURBO-FREEZER XL / XE in this application.

2.2 Saving frozen programs

The functions [S] , [R] and [F] will save a frozen program to external devices, like cassette or diskette (using [S]) or to the ramdisk (using [R]) or to the freezer ram. (using [F])

When [S] is used, a dialog will enquire if the data will be saved to a bootable cassette or to a disk file. Before starting this function, the device should be ready for use. (Put a cassette or disk in the device) If a diskette is used, the next dialog will ask for a filename. This input can be edited the normal way.
Attention: If saving to boot is selected, data will be written do device D1: and any data on the disk will be lost!

If no filename is entered and only [RETURN] is pressed, the filename "CORE" is used. Special: If an existing file is to be rewritten, wildcards can be used. And if no D: ; D1: or D2: is entered, the default is used. D1:. The freezer will support a maximum of 8 disk devices. (D1: up to D8:)

[S]	Save frozen program to diskette or cassette.
[R]	Save frozen program to the Ram-disk.
[F]	Save frozen program to the Freezer-Ram.

During the write process, the menu may switch of and on, this is normal. And to enable maximum speed, the screen may be jumpy, a small price to pay for the speed. Without these imperfections, it would take twice as long to save the data to the ram-disk.

2.3 Clearing the OS-RAM

As there are an increasing amount of programs that need 64K of ram (and also use it) the freezer has to accommodate this as well, so the area behind the OS is also useable. But during boot-up, this area is not cleared, so this useless data will take up space on the external data sources. To prevent this, it's possible to clear this area from $C000-$FFFF using the function [Z] if the program only uses 48K. This will result in a decrease of the amount of data with 25% to 50% and consequential shorter loading times.

[Z]	Clear ram behind OS.

This function can be used before booting a program or before saving the data. The first option is the preferred one, when in doubt if the area is in use.

2.4 Starting DOS and the debugger

Activating the [D] will start the DOS and debug function. This function uses a full screen and a command line, as menus will not suffice for this part. However no Atari ram is used for this function. And the content of the frozen hardware registers is also unchanged.

[D]	Start Debugger and DOS

2.5 Backup of programs with reloading parts

Freezing programs that load more data when running is no problem. But once the program is resumed, the disk that contains the additional data must be available to the program. And of course, this disk will be used. Because of this, a backup of the disk would be very good, so it can be stored on a safe place. Using a drive with a speeder is not always an option, as many programs have copy protections.

Because the freezer doesn't remove any protection from original disks, the disk, preferably with write protection, should be present in the drive. If the game needs some saving on a disk, make sure the right one is used. So don't forget to replace the disk that contains the frozen data. If possible, use a backup of the original program for the experiments and put the originals in a safe place.

3 The built in DOS

Who didn't encounter the following fathom: for the last 3 hours, a program was edited and now it's time to save it again. But instead of the correct system response, the message "File locked" or "Disk full" is displayed. And MEM.SAV was not in use, because this takes for ever to execute, so calling DUP is also not an option.

As the freezer is capable of stopping any program at any point, it's very tempting to also include a DOS function with the most commonly used commands. And then it's no problem to clear the situation and continue the program after this.

The DOS contained in the Turbo-freezer XL / XE has the services needed to solve the above situation. It's fully compatible to DOS 2.0 and 2.5 and supports single-, enhanced-, and double-density formats. And the functions of Turbo 1050 are fully supported, as well as the high SIO speed of the Happy / Speedy and XF551. But any other tuned or standard device can be used as well.

3.1 Enterng Commands

Command entry is done in the command line at the bottom of the screen. Editing inside this command line is possible in the usual manner. The cursor cannot leave the command line. Using the arrows for cursor up and down, the previous commands can be recalled. The freezer keeps the last 4 commands that were entered.

A disk command consists of a 3 character long command or a command followed by one or more spaces and a filename. Manny command also allow for an option, that may be placed behind the command, separated from it by a slash. The command itself must be directly behind the prompt. And besides the space that is mentioned before, no additional spaces may be entered. Filename may contain the wildcard "*" replacing any string of characters, as well as the "?" to represent just one character. Any illegal command will simply be ignored and will not cause an error message. Error will only be displayed, when caused by the execution of the command.

3.2 Basic rules for the execution of commands

When the "D:" is not included with a filename, the default will be "D1:" And the ram-disk is not supported, as this depends on the DOS that's currently in use.

Wildcards may also be used in the destination file names. In this case, the first matching filename is used. The only exception is the rename command [REN], this will be described in the next part.

The commands used to manipulate directory entries, DEL, LOC, UNL, and REN may affect multiple file names in a row. To prevent unwanted actions, an option has to be added to have the command be effective on multiple files with similar names. The option "/Q" will ask for confirmation with "[Y]" for every matching file name and any other key will not be accepted. Stopping is possible using the [BREAK] key.
And if you're sure, use the option "/A" to include all files with matching names.

Errors will be displayed in readable text and will result in a return to the main menu.

Command summary

3.3 Command Summary

The commands are listed here in logical order, as they are not new and unknown. Beginners may lookup the commands in a DOS manual, like the one that came with the 1050 drive.

DIR	Show directory
DIR Filename	Show particular filename
DIR Filename	Delete filename
FMS	Format single density
FME	Format enhanced density
FMD	Format double density [1]
LOC Filename	Lock file
UNL Filename	Unlock file
REN Filename, new filename	Rename file
LOA Filename	Load object file
SAV Filename, Hexadr1, Hexadr2	Create file (save from Hexadr1 to including Hexadr2)
SAV Filename/N, Hexadr1, Hexadr2	Create file without COM header. (Raw data)

[1] Formatting in double density is not possible with the standard 1050. But this may be done using a 1050 Turbo, the cheapest way to get this amount of data storage.

3.4 Specialties of several commands

Some commands have specific characteristics compared to standard DOS commands, offering improvements or are the logical result of the freezer environment.

While renaming files (REN) wildcards are allowed. The wildcard characters are replaced with the characters from the original filename. This feature allows for the operation to effect groups of filenames in a more efficient manner and is not restricted to primary or secondary group criteria.

Object files can be loaded using LOA. The destination will then be displayed. (More than one for compound files) The destination may be the total 64k address space. This includes the contents of hardware registers (that are frozen at the time) The program will not be resumed, to prevent conflicts that may arise when the memory management of the freezer loads multiple fragment to the memory.

3.5 DOS error messages

Using the DOS may cause various errors. (The load and store functions of the freezer for frozen programs also use DOS) The error messages are in readable text and in English. This is done to ensure that most people will be able to understand the messages and translation is not needed.
The errors are listed below:

`FILE NOT FOUND`	File is not loaded
`FILE# MISMATCH`	Internal file structure is corrupt. Use salvage program.
`BAD DISK I/O`	There's a drive error or write protect, preventing the command execution
`NO DRIVE`	No device response
`DISK FULL`	
`FILE LOCKED`	
`DIRECTORY FULL`	The maximum of 64 files per disk is reached

4 The Debugger

A debugger is used by machine language programmers to find errors in the object code, directly in the computer memory and to alter (and improve) memory contents. In combination with the freezer, there are certain advantages. Because of the way the freezer works, there are also some limitations, that may be eluded by using the proper way to work.

The biggest advantage without a doubt is the possibility to work with the frozen system contents. Doing this, creates the impression of having a second computer, that's linked into the first Atari, enabling a "look behind the screen", where the system can be restarted at any time. A debugger without freezer will have the big problem of it's own memory usage and the actions of the debugger itself that influence the system status. A programmers name for this is "Trashing", meaning turning the stuff into waste.

A standard debugger will use Ram to operate and may trash the data that was put there by the program that's being looked at. And even if the debugger has it's own Ram, it will still trash the hardware registers of ANTIC, POKEY and GTIA. The shadow registers in the OS do ignore the player missile graphics and the sound registers and will be deactivated by many programs. And as a result of the trashed hardware registers, it will be a lot of work, if even possible, to resume a program. And apart from this, a debugger without freezer doesn't have the ability to view the (often not readable) hardware registers. So even if the register contents is not trashed by the debugger, the values in the registers remain unknown.

Using the debugger in the Turbo freezer XL / XE, all information about the system is available. The contents of the hardware registers (What has been written there, not the status returned by a read operation) can be viewed in the I/O area and may be changed, without having to fear a system crash. The changes are made only in the frozen program status. And it's also without penalty to use the total Ram area (Even behind the system OS-Rom), so no trashing or crashing to fear.

Added to this, are various advantages created by the environment of the Turbo freezer XL / XE. The load and save function allow for instantaneous testing and retrying, without lengthily reloading a program. And if the change was not as expected, it may be undone in seconds. Using this, even horrendous code monsters that disable DOS or reuse the memory occupied by Dos and cannot be accessed the simple way, will have to give in to the powerful features and thus are not frightening anymore.

There's also some disadvantages. The frozen situation is operated on by the freezer. For this reason, any action that is done, will only have a visible result after resuming the program. In case of a ram disk, it will be hard to use the freezer for debugging. The program will be in full action when resumed and it's very hard to resume a program at a different point, this will ask for some ingenuity and fiddling with the hardware registers.

Some limitations are caused by the amount of available memory. For example a look into the OS or the direct creation of compound files. If this is a disadvantage, is mainly dependent on the working style of the user. The author prefers not to chase the OS by using a debugger, but instead use OS-listings and use small portions instead of a big code monster, as is the nature of a compound file. And at the end of the day, it's quite simple to use a DOS to combine all parts into one compound file.

4.1 Debugger commands

As the functions are quite straight forward and based on the Atari editor assembler cartridge, similar to many other debuggers, there’s only a summary made into a table. The commands may not contain any spaces and must start immediately behind the prompt. The entry must fit in the command line. All entries are in Hexadecimal. By omitting a value, memory locations or registers may be left unchanged.

Example: C100<0A,,4D will change $100 to $0A and $102 to $4D. After this, the internal address counter (PC) will point to $103. The contents of $101 is unchanged.

For all commands, the address may be omitted and the debugger will use the internal address pointer or a value that will give a correct representation. The output can be stopped at any time by pressing [S] (stop) and resumed by pressing [Q] (continue)

D Adr1, Adr2	Display memory contents
I Adr1,Adr2	Display memory contents in hex and ascii (Display Internal)
Q	Return to main memory (Quit)
R	Display registers
R<Byte, Byte	Change registers
L Adr1,Adr2	Disassemble (List)
DL Adr1, Adr2	Display List Disassembler
G Adr	Change return address (GO)
C Adr<Byte, Byte	Change memory contents
SR Secno	Read Sector
SW Secno	Write sector
SIO Ö	Execute SIO command
/Adr1,Adr2/Byte	Search function
V	Show interrupt and V-count at moment of freeze.
V<Byte	Change VCOUNT
VER	Display freezer software version number
DIO<Byte	Activate direct IO mode. Use with caution.
A Command	Atari SIO remote controll

Remark: With the exception of the spaces behind the commands, all spaces in the above list are only there for readability and should not be entered with the command. The PC cannot be changed using R<, the software doesn’t allow this. Use the G command for this. The sector buffer for the SW and SR command is the same as the buffer that DOS uses and the memory-manager will remap this for the freezer to $D700, where it may be edited using the debugger. The sector is not physically present here! To use another device as D1:, simply put “Dx:” before the sector number.

Addresses and Bytes will use hexadecimal format. To use decimal format, simply precede the byte with the percentage sign (“%”) Example: “C%710<%10 is identical to “POKE 710,10” in Basic. Bytes may also entered in ATASCII or internal Atari screen codes. For ATASCII, precede the byte with an apostrophe (“’”) and for screen codes with the AT-sign (“@”)
Example: “C0600<‘H,’a,’l,’l,’o” or “C9C40<@S,@c,@e,@e,@n”

Specially for the use of vectors (Like the Display-list or interrupt vectors) there’s another variation to enter an address: When an address is preceded by an asterisk (“*”) then a (2byte) address is taken

from the address that is pointed to and the contents of this location is used. This makes it possible to read the data from the current display list.
Example: “DL*230” or “DL*%560”

The display list disassembler displays the ANTIC commands as mnemonics (As described in the ANTIC data sheet) “BLK x” means x spaces, “CHR x” means text mode x, “MAP x” means Graphic mode x, “JMP Adr” means jump to address, “JVB Adr” means a jump after the vertical blank interrupt. Apart from these mnemonics, the following options, if applicable, will also be included: “LMS Adr” means Load Memory Scan counter, “H” and “V” mean Horizontal and Vertical scrolling respectively, “I” means execute Display List Interrupt.

The search function may used in various manners: “/ADR/Byte1,Byte2Ö” will look for the byte sequence from the given address. Once the sequence is found, the search stops and the address is displayed. The internal address pointer is also set to this address, so it is then possible to use the “L” command to disassemble the data on this address. Using “/” the search can be resumed. The sequence may contain 8 bytes, this is sufficient for most cases. It’s also possible to omit bytes from the sequence, these are then ignored. “/1000/8D,,D4” would result in the first address started from $1000 that does a write to the ANTIC (STA $D4xx) The search bytes may also contain a bit mask. To do this, the byte should be followed directly by “&” and the bitmask. Example: “03&0F” The bitmask is and-ed with the search byte first, and the result is compared with the memory.
If for this, also a start and end address is entered, “/Adr1,adr2/Byte1,Byte2Ö” the given area is searched and all addresses containing the search bytes are displayed. The search doesn’t stop when the first occurrence is encountered. Many times, the complete memory is to be examined and there’s a very short way to do this: “//Byte1,Byte2..” is the same as “/0000,FFFF/Byte1,Byte2..”
If the sequence is long or the memory area is big, the search may take many seconds. Using the [BREAK] key, the search can be interrupted at any time and [/] will cause a resume.

Using the “SIO” command, any SIO-Command can be given, just like using the $E459 SIO-Vector) The memory address (DBUF) is set to the vector buffer address ($D700) and the length (DBYTE) can be a maximum of $0100. The “SIO” command must be followed by the values for Devise, Unit, Command, Direction, Timeout, Length, Daux, with the same significance as that of the memory locations $300-$30B in the Atari. The SIO command retains the latest values, so in the next commands some parameters may be omitted. If no parameters are entered, the last SIO command is repeated. The default setting is “Get status from D1:” to make sure that no serious things go wrong when just “SIO” is entered. One should use this command with caution and check for the correct parameters, to make sure that the wrong disk will not be formatted!

The “V” command will return the type of interrupt (NMI or IRQ) that caused activation of the freezer, together with the original (this address is rerouted to the freezer software) interrupt vector address and additionally also the value of VCOUNT at the moment of the freezer activation as well as the calculated value at the moment the interrupt occurred. This command was mainly introduced to test the freezer firmware and may be of little interest to the average user. But here’s a description of the significance and origin of the values: When the freezer is activated, the freezer software Rom is mapped in to activate the program and the High-byte of the interrupt vector is rerouted so the freezer software is activated. Because the low byte was not changed, there’s a full page of NOP instructions in the freezer software. It may be that the Atari has to get through many NOP instructions to get to the part that will read the value of VCOUNT. Using the low-byte of the interrupt vector, the software will try to calculate the amount of time it took to get trough the NOP part and adjusts the VCOUNT value accordingly, in order to get the exact moment that the freezer was activated and should return to for the resume. This calculation is not 100% correct and may

vary according to the graphic mode. In the very rare case that the value is not correct and causes problems, the command “V<Byte” will set the VCOUNT to the desired value by hand.

Changes to the hardware registers will only be done after a resume and not instantaneous as with the standard debugger. After all, the frozen memory contents are manipulated.

In the Freezer 2005, the possibility to switch this of is created for professionals, the direct IO mode. The command “DIO<1” switches it on, “DIO<0” deactivates it and “DIO” gives the current status..

ATTENTION: Using the direct IO mode switches of part of the freezer functions and one could end up shooting his own foot! The freezer will not notice any changes done in the direct IO mode, therefore one should do the same changes with the direct SIO mode deactivated, to make sure that the registers are not set back to the original state by the resume command.

Using the “a” command, remote control commands can be sent to ATARISIO. This enables a total remote control of ATARISIO without any additional program. More information can be found in the ATARISIO manual.

Using the “VER” command, the current version of the freezer software is displayed. If there’s no version number in the current Flash-Rom, the result will be “n/a” on the display.

5 The Oldrunner (the old ATARI OS)

The Atari is the only 8-bit home computer class that contains a well designed and structured Operating System. This allows for changes, extensions and improvements without the need to adapt current programs. That's why Atari could use a more sophisticated OS in the later XL series.

Despite this, still there are programs (from 1980-1983) that don't run on the Xl series. This is the fault of the designer that didn't consider the official programming guidelines.

To be able to use these programs on the XL machines, Translator disks were created, that disable the OS in the machine and use the Ram behind the OS to store an alternative OS. Although this works well in most cases, it's time consuming and unpleasant to always have to load this alternative OS. The better solution is to have the alternative OS available in a ROM, so it cannot easily be altered and doesn't have to be loaded all the time. This solution works for all programs that don't work with the XL bus used to involve opening the computer and paying a lot of money for it. (Prices up to $50)

Using the TURBO FREEZER XL/XE it's also possible to have this feature, without opening the Atari. The memory management logic makes it possible!

There's a switch on the TURBO FREEZER XL / XE marked with "OS" to enable the oldrunner. For the XL version, the switch is located towards the middle, for the XE version towards the PCB edge. To activate the Oldrunner, the XL switch has to be moved in the direction of the PCB side, the XE switch has to moved away from the Atari. It is best, not to move the switch with the Atari switched on. (The temporary Ram locations by both OS-es differ and therefore the Atari will crash)

XL: Switch "OS" (In the middle of the PCB) toward the PCB edge = "Olrunner on"
XE: Switch "OS" (Near PCB edge) away from the Atari = "Oldrunner on"

For various reasons, it's best to only use the "oldrunner" mode if this is the only way to get a program to work. The oldruner mode has no on-board basic, no ramdisk and no warm boot (e.g. non destructive RESET) Using the [RESET] key, the oldrunner always executes a cold start. The key on the old Atari was marked [System reset] and caused an NMI. It's not very difficult to retrofit this key (See 1.4) but it involves opening the Atari and this should only be done by experts. This key is not absolutely needed anyway, as most incompatible programs are games that will do a coldboot or will have some crash programmed in case of reset. This kind of reaction was used in a petty attempt to agitate "Crackers" but only serves to annoy the users now.

6 The Cartridge Emulation

The cartridge emulation offers another very powerful feature to the TURBO FREEZER XL / XE user. This is capable of emulating: standard 8K, standard 16K, 16K OSS bankswitching (16K modules using only 8K like Mac/bug65, Action, Basic XL or XE) and even the 64K Sparta-dos. (A 64K module that uses only 8K)
The cartridge emulation can use all free memory of the Flash (480k) and also all Ram (112K). This means a total of 60 cartridges can be available all time and the Ram can be used for experiments with cartridges.

The cartridge emulation is not only a very good feature for all who use many different cartridges and grow tiered of changing the cartridges all the time (This is also better for the cartridge port), but also form developing cartridge based software. Because the data can be stored in both the Flash-rom and the Ram, changes in software can be executed and tested very fast.

At http://turbofreezer.horus.com/ an adapted version of the atarimax maxflash software is available. With this, most disk images and com/exe files can be converted into module images. A special feature is that the TURBO FREEZER XL / XE also supports more than one disk image at the same time and the rest of the Flash-rom can be used to emulate other cartridges. This makes the TURBO FREEZER XL / XE much more flexible than the AtariMax MaxFlash module.

To activate the Cartridge emulation, the switch “CA” must be set to enable the emulation mode. For the XL version, the switch (or jumper) is near the side of the PCB. The XE version has the switch (or jumper) near the centre of the PCB.
The emulation is active if the switch is closed. (or the jumper is set) For the XL this means the switch towards the side of the PCB, for the XE the switch should be away from the Atari.
XL: Switch “CA” (Side of the PCB) towards the edge = “Cartridge emulation on”
XE: Switch “CA” (Middle of the PCB) away from the Atari = “Cartridge emulation on”
Just like with the oldrunner, only change the switch with the Atari switched of, otherwise the Atari will most likely crash.

When the cartridge emulation mode is active, a small menu will appear that enables configuring all settings of the emulation. The newly delivered TURBO FREEZER XL /XE has an (almost) empty Flash Rom, the only software in it is the Freezer software and the system disk writer. In the following part, there’s a description on how to program and use the Flash-Rom and the Freezer-Ram for cartridge modules.

The menu offers the following configuration options:
Using the [M] key, the cartridge type can be selected, form 8K, 16K, OSS, SDX, PDOS and OFF. “PDOS” can also be activated with [P] and OFF using the [O].
PDOS offers the full MyPicoDos and activating this automatically deactivates the cartridge emulation. This makes it possible to start a powerful Dos game without an extra disk or bootfile. “OFF” means the cartridge emulation is disabled.
Below “BANK” the current bank number can be seen. The keys [0] to [0] and [DELETE] can be used to select a decimal bank number between 0 and 59.

The [S] key switches between Rom and Ram as source for the emulation image.

The next option is the “KEEP (settings)” and can be switched using [K] between “YES” and “NO” and determines what will happen when the [RESET] key is pressed.
With keep as [YES] the settings will be saved after a reset, [NO] will reset all options and return to the cartridge menu.

The [B] is the last one and determines if a cold or warm boot will be executed to start the cartridge. The normal setting is “COLD” to make sure that the cartridge will start correctly. The setting “WARM” is usefull when “KEEP” is set to “NO” and the selection menu is activated again. With “BOOT” set to “WARM” the data will stay in the memory and the options can be set again, so a normal Reset will get the normal operation started.

When all is set correctly, pressing the [RETURN} key will activate the selection.

[M]	Change mode 8K, 16K, OSS, SDX, PDOS, OFF
[O]	Select the Off mode
[P]	Select PDOS
[0]-[9],[DELETE]	enter bank number
[S]	Ram/Rom for cartridge emulation source
[K]	Keep settings YES/NO selection
[B]	Boot COLD/WARM selection
[RETURN]	Start cartridge emulation

To use the cartridge emulation, a couple of important facts should be taken into consideration to fully understand the functions: The Flash-Rom is internally set up as 8K banks. And with a maximum of 512K, there’s a total of 64 banks with 8k size. (Bank number 0-63). The upper 32K are used by the freezer software and cannot be used for cartridge emulation. This leaves bank 0 to 59 to be available for the emulation. The Ram reserves the upper 16K for internal use so this leaves bank 0-13. Making a Snapshop will store the data in the lower 64k, so this then leaves bank 8-13 for cartridges.

When putting data into the Flash-Rom (and the Ram as well), there’s some more to consider: Standard 16K modules and 16K OSS modules have to start at an even bank number (At a 16K border) and the Sparta-Dos modules have to start at a number that can be divided by 8 (at a 64K border)

The emulation has this name for one more reason as well. “Real cartridges” have some lines in the cartridge port that allow for information to the Atari, about the cartridge being there or not. But the TURBO FREEZER XL / XE is connected to the PBI and lacks these lines. This is usually no problem. The only exception is cartridges that can completely switch off a cartridge. The OS does a compare of TRIG3 ($D013) and GINTLK ($03FA) in the VBI routine. When these values are not identical, this means a cartridge has been installed or removed and the OS will enter an endless loop. The next reset will then cause a cold-boot.

A bankswitching module would have to also change the value in GINTLK if it wants to switch completely off or on. The simples way is to read the value TRIG3 and store this in GINTLK. But unfortunately, not all modules do this, some alter the contents of GINTLK directly into 0 or 1. But the value of TRIG3 is not changed by the cartridge emulation, so the Atari will lock up.

To enable these kind of modules to also work in the emulation mode, the original code has to be patched. The disk contains the program “PATCH.COM”to do this. It will patch the ROM-Images of

Basic XE and Sparta-Dos versions 4.19, 4.21 and 4.22 in such a way that they will work in the emulation. Basic XL, Mac/bug 65 and Action will work without any modification.

7 Programming the Flash ROM

The software of the TURBO FREEZER XL / XE is not stored in an Eprom, as is most usual, but in a Flash-Rom instead. The biggest advantage of this is that the Flash-Rom can be programmed by the Atari itself. This enables software updates at any time, as well as additions to the cartridge emulation in as jet unused parts of the Flash, without having to use a specialised programmer. The Flash-Rom in the TURBO FREEZER XL / XE can be reprogrammed as often as 1.000.000 times (According to the manufacturer of the chip) and this is sufficient for most experiments. And in case the one-million programming cycles is exceeded and the Flash is defective, the Freezer doesn't have to be repaired. The Flash-Rom is in a socket and can easily be replaced by a new one, an AMD 29F040. And if 128K is sufficient, the Eeprom rypes SST 29EE010 or Winbond 29EE011 are also usable. These chips are often used on PC-mainboards for the BIOS and can be removed from an obsolete board. (And the freezer can even be used to reprogram the BIOS chip in case something went wrong with upgrading the PC-Bios) At the moment, the AMD29F010 is also supported and more types can be made available by a minor software change. (Or use a programmer to store the data in the chip. Only the programming software of the Atari checks for the Flash-type, the freezer software doesn't check)

For normal operations, the software in the Flash-Rom of the TURBO FREEZER XL / XE is write protected. To enable writing, the jumper marked “FL” should be closed. (FL stands for “Flash write”) The jumper is located between the switches for Oldrunner and cartridge emulation. When it's open, both the Flash-Rom and the Ram are write protected. If the pins are connected, writing to the Flash-Rom and / or Ram is enabled.
XL/XE: Jumper “FL” (Between switches) on = “writing to Flash-Rom and / or Ram enabled”
It is highly unlikely that any software will write something into the Flash-Rom.
To store data in the Flash-Rom, a special sequence has to be used to access the Flash-Rom, if this is not done, a write will simply be ignored. For this reason, it's not dangerous to have the jumper on the pins all the time. But if you want to be on the safe side, pull the jumper after programming the chip.

To program the Flash-Rom or Ram, connect the pins with the jumper and start the FLASH.COM software on the TURBO FREEZER software disk. If the pins are not connected by the jumper, the program cannot find the Flash-Rom and displays an error message. In this case, just put the jumper in place and answer [Y] to the query “restart program?”

The AMD29F040 Flash-Rom has an internal makeup of 8 blocks, 64K in size. These blocks can be individually reprogrammed. This means that programming a part into the cartridge emulation will not influence the freezer software (And vice-versa)
This configuration with the 64K Blocks has a minor disadvantage as well: It's not possible to erase a smaller portion than 64K. (Programming would be possible per byte though) But to keep matters simple, a 64K block should be compiled beforehand. So a set of 8 blocks of 8K cartridges should be programmed in one go. This limitation is not applicable to the Ram, here any 8k (or smaller) block can be programmed individually.
The image file may be 512K maximum size. The limitation here is that the size should be a multiple of 8K. However if the image file is only 16K (for example) then the other 48K of the 64K block will be erased.

Once started, the Flash program will display the following data:

- Flash-Rom type.

- Bank size. (8K or 64K)
- Actual version number and production date of the freezer software.

In case the Flash-Rom doesn't contain the freezer software, "n/a" is displayed instead of the version number. Using the "VER" command in the debugger, the version number can also be displayed.

The Flash program menu offers the following options:

1) Program Flash-Rom	Program one or more blocks of the Flash-Rom
2) Write Flash ROM to file	Contents of Rom will be written to a disk file
3) Erase Flash ROM	Erase the complete Flash (Including the freezer software)
4) Program Freezer RAM	Store data in the freezer Ram for cartridge emulation.
5) Write Freezer RAM to file	Contents of RAM will be written to a disk file
6) Run cartridge	Start the cartridge selection menu from the freezer rom.

In general, the following is done when the Freezer Rom or Ram are to be programmed:
First, the Bank number is requested. This must also be a block boundary, the actual block size is displayed below the Flash-type.
Second, the filename of the Rom-image must be entered. This Rom-image may only contain the data to be programmed (no COM-header) and the file size has to be a multiple of 8K.

Now, the data is programmed to the Flash-Rom (or Ram) in 8K multiples, until the end of the image-file or the final bank number.

In case the freezer software is (by accident) erased and the freezer doesn't work correctly, or in case a new version of the software must be programmed, act as follows: The Flash-Rom data from the file FREEZER.ROM must be programmed into banks 56-63. So enter "56" for the bank number and "FREEZER.ROM" for the filename. As simple at that. And if there's a 128K Flash in the freezer, use the file FRZ128.ROM into the banks 12-15.

The current Freezer software is 32k in size, this is the second half of the FREEZER.ROM (Banks 56-59 are empty) Using a simple trick, this part of the memory can also be used for cartridge emulation. Just create an image file that contains the cartridge data in the first part of the image file. The second part should contain the freezer software. (Use either FRZ128.ROM or the second half of FREEZER.ROM)

7.1 Remarks for the adapted Maxflash software

The complete manual for the AtariMax Maxflash software can be found at: `http://www.atarimax.com/` Only the changes for TURBO FREEZER XL / XE are in the next part. The original software cannot be used with the Freezer, but the adapted version will support both the Freezer and the AtariMax Maxflash Cartridge.

For the software installation: Instead of the TASM Assembler, that's only available for DOS/Windows, the adapted software uses the ATasm for assembly.
The ATasm software has free source code and works both with Windows and Linux. So finally the Maxflash software can also be used with Linux. The ATasm version 1.05b (Or newer) should be installed, older versions have not been tested.

Specially for the TURBO FREEZER XL / XE there are some new options:
```
-FRZ32KB  -FRZ48Kb  -FRZ64KB  -FRZ80KB  -FRZ96KB  -FRZ128KB
-FRZ192KB -FRZ256KB -FRZ320KB -FRZ384KB -FRZ448KB
```

Have fixed cartridge image sizes. One of the options must be set! The size (32/64/..)KB stands for Kilo-bytes. The values for the MaxFlash Cartridge (1MB,4MB,8MB) were Mega-bytes.

To store more images into the Freezer, the bank number from where on the data has to be stored, must be explicitly entered. The default is using the first bank (bank 0). The option -FRZBANK=x is used to set another value.

And now two examples:

```
perl maxflash.pl -EXEPACKER -FRZ64KB test.bin ../test/
```

This will make a 64K image for freezer bank 0-7. Using FLASH.COM, the image “test.bin” can now be programmed into the Freezer, starting at bank 0, either to the Flash-Rom or the Ram.

```
perl maxflash.pl -EXEPACKER -FRZ64KB -FRZBANK=8 test.bin ../test/
```

creates a 64k image for bank 8-15. So program this to bank 8 and up.

The next options are not supported for the TURBO FREEZER XL / XE version of the MaxFlash software, as they are of no use.

-BIN2ATR (U)	Make existing BIN file into Flash image.
-BIN2ALL (U)	Make existing BIN file into CART/ATR image.
-FLASHER (*)	Make bootable ATR image for programming MAXFLASH cart.
-8MB (*)	Create image for 8Mb cartridge (Default 1MB)
-NOBIN (*)	Delete raw binary after processing.

In the cartridge selection memu, the MaxFlash images have the 8K type and bank must be set to the correct start number.

8 For Experts: Technical Details

The heart of the freezer is the CPLD that contains the complete logic for the freezer and cartridge emulation. One part is connected to the Atari, the other controls the Flash-Rom and Ram of the freezer. The switches, jumper and Freeze-button are also connected to the CPLD.

From the hardware side, both Flash-Rom and Ram are divided into 4K blocks. Usually, Ram and Flash-Rom are used in 8K blocks, but there are two cases where the actual 4K is used: For emulation of the OSS Bank switching cartridges and when the freezer is active and the freezer software accesses the freezer Ram. In all other cases, the logic only allows for access to 8K bank size.

There's yet another special case, while shadowing the hardware addresses of the Atari. (like ANTIC, GTIA, more of this later) The I/O chips don't have a full address decoding and the freezer has to cope with this as well. So the CPLD logic has a separate part to do this. For hardware shadowing, the address lines A4 and A5-A7 to Ram are inactive to the Ram. For other usage, these address lines are passed to the Ram, for Hardware shadowing they are deactivated.

When the Ram or Flash-Rom from the freezer have to be swapped into the Atari memory map, the internal Ram or Rom have to be switched off. This is done using a simple trick: The freezer pulls a line (That was intended as an output) the refresh-pin to low. And this lets the Atari act as if the Antic is executing the memory refresh, switching off the internal memory.
According to Berhard Engl (Who is a chip designer nowadays) the NMOS-Depletion Load technology doesn't get damaged when a pin is pulled low externally, while internally it's at a high level, because this is done by NMOS in this fashion anyway. If the chips had been in CMOS, this would not have been possible without damage to the ANTIC and then it might never have given a Freezer in the first place. The first attempts to create a freezer (1984) were based on rebuilding a complete Atari (It was actually done like this) with alternative select logic, because this trick was not yet know. It became 1985. Then the Refresh-trick was discovered, the Atari 800XL with it's parallel bus was sold in sufficient quantity and the brand new Gal chips allowed for the extensive logic to be compressed into a small number of chips. Alls this was needed to be able to create the first freezer and market it. It was 1987 then. Three years after the first tests. From then on, it took more than a year to develop the product (Not a man-year though, studies of Electronics were also done in this period) In comparison: it took only 3 man-years to develop the 6502 microprocessor. (Personal correspondence between Bernhard Engl and Bill Mensch) The processors in a modern PC took man-years in the thousands to develop.

So now to a description of the sole basic function of the Freezer: The freezer-logic and the cartridge emulation are implemented independently. Only at lets call it the end of the process, where the decision is made to use Ram or Flash-Rom and deactivating the Atari internal Ram/Rom, the two parts cooperate. This means the Freezer logic can actually also be started from within the cartridge emulation and it's possible to Freezer-Debug an emulated cartridge.

8.1 Details of the cartridge emulation

The basic functions of the cartridge emulation are governed by the jumper for Flash-write and the Cart-emu switch. If both the jumper is not installed and the switch is open (off), the cartridge function is fully disabled. This makes it possible to have a standard cartridge in the module slot (Or on the XE version) and use it without a conflict with the emulated cartridge.

If the Flash write jumper is on the pins, the cartridge emulation is not activated when the Atari is switched on. This jumper only enables the write access to the Flash-Rom and Ram and swapping this part of the memory into the Atari memory. This is used by the Flasher software.

To enable cartridge emulation from the Ram and to better protect the Flash-Rom, the programming facility has to be explicitly activated by the software. At every new start (or after pushing the [RESET] key) this write enable is explicitly deactivated again.

If at power on the emulation switch is closed, the cartridge emulation is activated. This is not affected by the status of the “FL”, flash write enable, jumper. If both are closed at power up, the emulation is active and writing is also possible. If the cartridge emulation is active, without the “FL” jumper on, only read access to the Flash-Rom or Ram are possible and write access is not possible.

There’s a small circuit on the freezer PCB that triggers a pulse at the power up of the Atari. Also, the reset-line of the Atari is connected to the CPLD. This enables the freezer logic to determine between a power up of the Atari and pressing the [RESET] key.

At power up, with activated cartridge emulation switch , the following events occur: The cartridge emulation is activated, cartridge type is set to 8k, Flash-Rom is set as source, Write access to Flash-Rom and Ram is disabled, the logic is set to maintain the settings after a reset and the bank number is set to 62. In this bank, the cartridge menu is stored, so this will be activated.

The default is for the cartridge logic to not change the settings after a reset. If “KEEP (settings)” is switched “OFF” the registers will be cleared after a reset and the menu will be activated just like it would when the Atari is restarted. This allows for testing a lot of cartridge emulations without having to switch the Atari on and off all the time.

8.2 Software configuration of the cartridge emulation

If one or both Cartridge emulation or Flash write is activated, the cartridge emulation can be configures by accessing the Address area $D500-$D5FF.

The following addresses are used for this:

$D580	Deactivate cartridge emulation.
$D581	Activate Cartridge emulation.
$D582	Pressing [RESET] will set defaults (Keep = off)
$D583	Pressing [RESET] will not set defaults (Keep = on)
$D584	No write access to Flash/Ram
$D585	Write access to Flash/Ram
$D586	Cartridge emulation from Flash-Rom
$D587	Cartridge emulation from Ram
$D588	Mode 8K Cartridge ($A000-$BFFF)
$D589	Mode 16K Cartridge ($8000-$BFFF)
$D58A	Mode OSS Bank switching Cartridge (16K in $A000-$BFFF)
$D58B	Mode Sparta Dos X Cartridge (64K in $A000-$BFFF)

$D540-$D57F	Bank number 0-63 ($D540 = Bank 0, $D541 = Bank 1)

Emulation of 16K Cartridges and OSS Cartridges is only possible for even bank numbers. More exactly, the LSB is ignored for the bank number and A13 is used instead, for OSS cartridges the A12 and A13 have their own bank switch logic:

In the area $B000-$BFFF the OSS cartridge always uses the first 4k block (Number 0), the area $A000-$AFFF is determined by the following addresses:

$D500	4K Block number 1
$D509	4K Block number 2
$D501	4K Block number 3

By accessing $D508, the OSS cartridges can be (temporarily) switched off completely.

For the emulation of the Sparta-DosX cartridge, the lower 3 bits of the bank number are ignored and the following addresses are used for the bank switching:

$D5E0	8K Block number 7 active
$D5E1	8K Block number 6 active
$D5E2	8K Block number 5 active
$D5E3	8K Block number 4 active
$D5E4	8K Block number 3 active
$D5E5	8K Block number 2 active
$D5E6	8K Block number 1 active
$D5E7	8K Block number 0 active
$D5E8-$D5EF	Temporarily deactivate cartridge.

To configure the cartridge emulation, the settings should be done in the following order:
1) Keep settings to on/off ($D582/$D583)
2) Cartridge emulation active/not active
If cartridge emulation is active:
3) Source for emulation (Flash/Ram)
4) Cartridge type (8K/16K/OSS/SDX)

6a) For OSS Cartridge activate block 1
6b) For SDX, activate block 0

To enable write access, the address $D585 should be accessed. When writing is finished, it's good practice to disable writing by accessing $D584, to prevent accidental destruction of data.

8.3 The Oldrunner Mode

If the oldrunner mode is active, the following happens: A read access to $E000-$FFFF will in reality access an 8K area at bank 63 (The last bank) in the Flash-Rom.
An access to $C000-$CFFF will not access any freezer chip and also disable any Atari access to memory. This simulates a memory hole that was present in the 400/800 models as well.

8.4 The Freezer Mode

As long as the freeze-button is not pressed, the freezer is invisible to the Atari. This means that no software can detect the presence of the Freezer.

The internal freezer logic has 4 states:

- Freezer inactive
- Freezer half active
- Freezer active
- Freezer temporarily deactivated

After power up, the freezer is in the deactivated state.

As mentioned before, in the freezer mode, the memory is divided into 4K banks (instead of the usual 8K banks) There's a total of 32 banks available.

To be able to have read access to the normally not readable registers at $D000-D7FF there's a trick in operation: Any write access to $D000-$D7FF will also write the data to bank 31 (The last bank) of the freezer Ram, as long as the freezer is in the inactive state. The Atari has no notion of this.

For a write access to $D2xx, $D3xx and $D4xx, the address lines A4-A7 to the Freezer Ram are deactivated. This is done because the address lines to the I/O chips in the Atari are not fully decoded either. A Write of $22 to $D400 followed by a write of $00 to $D410 will both end up changing the DMACTL register of the ANTIC, but the last value written (to $D410) is valid. And because the A4-A7 to the Ram are also not active, the write to both addresses will end up at $0400 in the 4K Ram bank 31, here the second write also rewrites the first one. A write access to $D0xx will only mask A5-A7 as the GTIA has a 32Byte area. ($D000-$D01F)

8.5 Activating the freezer

Once the freeze-button is pressed, first nothing happens. Only if the area $FFF8-$FFFF is accessed (This is were the IRO and NMI vectors reside) and the freeze-button has been pressed, the freezer goes to the half active state.

At the next clock cycles, there's a couple of possibilities.

If the Freezer-button is not pressed any more or another area than $FFF8-$FFFF is addressed, the freezer goes back to the inactive state.

But if the freeze-button is still active and another access to $FFF8-$FFFF is done, the freezer goes to the active state.

The ROM/Ram in the Atari is swapped out (As is the Oldrunner on the freezer) and Flash Bank 60 is swapped in. At the correct locations (Last 8 bytes of Bank 61) the new locations for the Interrupt vectors are present, those reroute the interrupt.

Once the Freezer has replaced the Interrupt vectors with it's own ones, the freezer goes to the active state,

And in active state, the following happens:
The freezer Ram is swapped into $0000-$1FFF. (More about this later)

From $2000-$3FFF the freezer-Rom is swapped in. At the beginning of the part, there's the alternative Interrupt routine, after this is the freezer software.

If the freezer software want to access the area $0000-$3FFF, the Rom and Ram of the freezer have to be temporarily swapped out again.

A Write access to $D700 will switch the freezer between the active state and the temporarily inactive state, back and forth.

To maintain compatibility with the existing freezer software and to enable simple use of the extended 128K Ram, the area was divided as follows:

The area $0000-$0FFF is always the last 4k bank of the Ram (So it's bank 31)
And the area $1000-$1FFF can be any 4K bank of the Ram. It's best to only use bank 0-30 here, as bank 31 is already in $0000-$0FFF.

The original freezer software only uses the area $0000-$07FF of the Ram, as it only had 2K of Ram available. This means that the area of $0800-$0FFF may be used for private extensions (Always available) and there's also an additional 31 banks in $1000-$1FFF.

Ram banks may be switched by accessing $D780-$D79F. $D780 is bank 0, $D781 is bank 1,..

The Flash-Rom is also divided into 8K banks. To make sure that the cartridge emulation and the freezer may work independently, the freezer mode has it's own bank select addresses. $D740-$D7FF $D740 is bank 0, $D741 is bank 1 and so on.

In the current version, bank 63 is used by the oldrunner, bank 62 contains the cartridge menu and bank 60 and 61 contain the freezer software. Bank 60 is the main freezer software bank. And because the original freezer Rom was absolutely full (So there was no room for extensions at all) the current software uses Bank 61 as well.

Future extensions of the software should use the higher banks to begin with, 59, 58. This way, the freezer is always in the to banks and the cartridge emulation from the bottom up.

If new software has to use more banks and should switch to another bank, it's good to notice that all 8K will be swapped. This means that after accessing $D740-$D7FF the next command will be in the new bank right on.

There's a trick to make the switching a lot easier:
Assuming the software is in two banks and one wants to switch from bank 60 and go to the routine "FOO" in bank 61. To the left, the routine for bank 60 and the right is bank 61.
First the code for the routines FOO and BAR

```
                          Bank 60                 Bank 61
$2A10      A9 10      BAR   LDA #$10                   . . .
$2A12      60               RTS                        . . .

$2E82                       . . .             A9 20            FOO     LDA
     #$20
$2E84                       . . .      60                RTS
```

The routine “FOO” in bank 61 is at address $2E82 and at address $2A10 in bank 60 is the routine “BAR”. From bank 61 the routine “FOO” can just be called using “JSR FOO” . This will then be assembled as “20 82 2E”
To switch to another bank, one can simply use the following “jumper board” to call with JSR:

```
                    Bank 60             Bank 61
$3FF0          8D 7D D7  SFOO STA $D77D 8D 7C D7  SBAR STA $D77C
$3FF3          20 10 2A       JSR BAR   20 82 2E       JSR FOO
$3FF6          8D 7D D7       STA $D77D 8D 7C D7       STA $D77C
$3FF9          60             RTS       60             RTS
```

A jump may look like this:

```
$3FFA          8D 7D D7  JFOO STA $D77D 8D 7C D7  JBAR STA $D77C
$3FFD          4C 10 2A       JMP BAR   4C 82 2E       JMP FOO
```

And here’s the explanation, what happens when “JSR SFOO” is called from bank 60:

First, from bank 60, at address $3FF0 the command “STA $D77D” is read and the executed. After this, bank 61 is active.

9 More information and Weblinks

Website about the TURBO FREEZER XL / XE with actual information, updates, design and source files and more:
`http://turbofreezer.horus.com`

Website of the ABBUC, where the freezer can be ordered:
`http://www.abbuc.de` (This is mainly in German language)

Lattice semiconductors: Datasheets for the iM4A5, Download of the design environment ispLever Starter:
`http://www.latticesemi.com/`

Advanced Micro Devices, AMD: Datasheets for the Flash-ROM 29F040:
`http://www.amd.com/`

10 Warranty and intended Usage

The warranty on material and assembly is 6 months. If the freezer is defective, it may be replaced by one that is functioning. In case of hardware incompatibility, the price of the freezer will be refunded, after the part has been returned, free of postage.
This warranty only covers the hardware. The risk for error free operation of the software and the application for specific tasks rest solely with the buyer or user. There are no rights, directly or implied, of whatever form, concerning software, exchange, improvements, refunds (Full or partial), changes, updating etc.

Warranty does not cover transport damages, incorrect installation or use, static discharges, pollution, attempted extensions and any alterations to the electronics outside our workshop. There's also no warranty to any parts that were not installed by us.

Usage of the TURBO FREEZER XL / XE must not in any way infringe rights of third parties or oppose any laws or regulations. Any responsibility or claims by third parties of any form, illegal use, damage to property or dead or living matter, and any other kind of claim is not accepted by us in any way and are the sole responsibility of the user. It's the sole responsibility of the user, owner or buyer to prevent illegal use or any other damage that may arise from using the product. Adults are responsible for their children.
The user should be aware of regulations or laws concerning the usage and seek council in case of doubt. Claiming ignorance or lack of knowledge is not an option out.

The use of commercial names, trademarks, labels or manufacturer makes is not done with the assumption that any of these items should me freely usable and no rights of the items are intentionally claimed or assumed. All referenced do not take any patents or other rights into account. Any claim from manufacturers is strictly excluded. Use at your own risk.

11 Reflections by the translator (and PCB designer)

This project has been quite a challenge. I've tried to not only translate the text in a correct way, so readable English would be the result, but also keep the "spirit" of the original (And the original text was in German and I'm Dutch). The manuscript has been proof read by a couple of people, but still some errors may have remained. Extra care has been taken however, to make sure the technical information is correct.

The total project would not have come this far and would not have evolved into the current state without the internet and E-Mail system.

During the design of the schematics, PCB, CPLD contends, prototyping and testing, numerous E-Mails were sent and discussions done.

And the ABBUC website was the place where I first became aware of the project.

This started me asking questions and I've kept on doing this all through the project. By doing so, a lot of new ideas came up and were included into the end product.

So in the end, there's a bit of a philosophical conclusion: Cooperation and most importantly listening to one another can lead to very good things.

Enjoy the freezer and keep on asking questions!

Ede, December 24th 2005.
Guus Assmann

P.S. If anyone cares to translate this manual into another language, please contact us. We'll be happy to see the results.